Ø4Ø2Ø

I Edizione

Gianni Becattini

RPN/8A2T

MANUALE DI UTENZA

Il presente manuale comprende tutte le informazioni utili per l'uso del linguaggio di programmazione RPN/8A2 su telescriventi di tipo Baudot a 5 bit, seriali. Non sono compresi nella presente edizione i list oggetto.

SISTEMI DI ELABORAZIONE - MICROPROCESSORI
VIA MONTEBELLO, 3 - 3a rosso
TEL. 055 / 219.143 — 50123 FIRENZE

I N D I C E

Ø4Ø17

I Edizione

Gianni Becattini

M A N U A L E   D I   U T E N Z A

Il presente manuale comprende la descrizione completa dell'RPN/8A, un interprete ad alto livello per i microcomputers della serie CHILD 8/BS

SISTEMI DI ELABORAZIONE - MICROPROCESSORI
VIA MONTEBELLO, 3 - 3a rosso
TEL. 055 / 219.143 — 50123 FIRENZE

1 - Scopi e finalità del linguaggio RPN/8A

Si é cercato di realizzare un semplice interpretatore per consentire una maggiore flessibilità di impiego di piccoli elaboratori basati su microprocessori. Tra gli obiettivi, quello di ottenere un interprete molto compatto in modo da consentire anche un conveniente uso da memoria in sola lettura (ROM), con possibilità di collegamento con programmi di tipo assembler in modo da facilitare l'interfacciamento di sistemi periferici.

Per il funzionamento dell'RPN/8A sono richiesti 2 K bytes di memoria RAM o ROM per l'interprete e 1 K di memoria RAM per il programma utente e per i dati.

Pur non trovandoci davanti alla versione definitiva si crede di poter affermare che la versione attuale consente già diverse possibilità fra cui non ultima quella di tracciare semplici grafici grazie alle funzioni di cui è dotata.

2 - Principali innovazioni rispetto alla versione precedente

La versione cui si riferiscono le presenti note (Vers. 28.8.77) differisce dalla precedente soprattutto per le più consistenti capacità aritmetiche di cui è dotata. Si è provveduto l'interprete di una aritmetica in virgola mobile con 14 cifre significative e segno comprendente le quattro operazioni. Sono stati aggiunti complessivamente 13 operatori. Il formato di stampa dei dati in uscita è prefissabile, mentre la precisione dei calcoli rimane in ogni caso la massima consentita. Come già accennato si è reso possibile il salto ad un programma di tipo assembler,

consentendo così all'utente di collegare la capacità operativa dell'interprete con l'uso più dettagliato possibile solo a livello di linguaggio base.

Un apposito operatore consente di accettare dati da tastiera in modo da facilitare la scrittura di programmi conversazionali. Gli operatori di decisione sono stati portati a tre, mentre un operatore di tabulazione permette di tracciare semplici grafici sul terminale usato.

La memoria programma è stata estesa ad oltre 750 bytes mentre lo stack a quattro posti di nuovo tipo controlla automaticamente e ciclicamente le condizioni di fuori capacità (overflow) e sottocapacità (underflow) e libera pertan-

to l'utente dal tedio di tenere conto dello stato del medesimo.

Per quanto l'interprete sia stato, rispetto alla precedente edizione completamente ridisegnato, si può senz'altro affermare che l'esperienza passata è stata determinante ai fini attuali.

In appendice è riportata una tabella che sottolinea nel dettaglio tutte le differenze tra l'RPN/8 e l'RPN8/A.

3 - Struttura dell'interprete RPN/8A

L'interprete è caratterizzato da due modi di funzionamento: il modo di INPUT in cui si introduce o si modifica il programma sorgente, ed il modo di EXEC in cui si esegue il programma memorizzato.

Corrispondentemente a questa dualità di operazione si possono riconoscere due parti fondamentali nell'architettura del sistema. Esaminiamo più in dettaglio i vari blocchi logici che compongono l'RPN/8A.

3.1 - Il modo di INPUT

In fàse di INPUT (la macchina stampa ":") ogni carattere battuto viene esaminato ed in base alla sua codifica ASCII si decide se introdurlo in memoria come facente parte del programma o se (carattere ";", eco "(") passare il controllo all'unità di decodifica dei comandi.

Una terza eventualità è costituita dal carattere "BELL" (eco " ") che serve per cancellare dalla memoria l'ultimo carattere introdotto. (fig.3.1.1)

Il decodificatore dei comandi (richiamato come già detto dal carattere ";") controlla se l'operazione richiesta appartiene all'insieme di quelle riconosciute ed eseguibili dall'interprete eventualmente segnalando l'errore con un ritorno al debug e passa il controllo alle opportune procedure (fig. 3.1.2).
Uno dei possibili comandi è quello che serve per passare nella fase di EXEC.

3.2 - Il modo di EXEC

In conseguenza del comando "e" il controllo viene passato all'analizzatore lessicale che, in base alla pri-

ma parte della codifica ASCII di ogni carattere del programma (vedi tabella 3.2.1), stabilisce la routine da richiamare, sospendendo temporaneamente la sua analisi. (fig.3.2.1)

Tabella 3.2.1

Codifica ASCII	Significato
2x, 5x,6x,7x,	Operatore
3x	Numero
4x	Variabile
OD	End of Record (Car.ret.)

4 - Descrizione operativa dell'RPN/8A

Una volta caricato l'RPN/8A in memoria (inizio esecuzione la 1000) la macchina stampa

```
* RPN/8A (vers.28.8.77)
:
```

Il carattere ":" avverte che ci si trova nello stato di

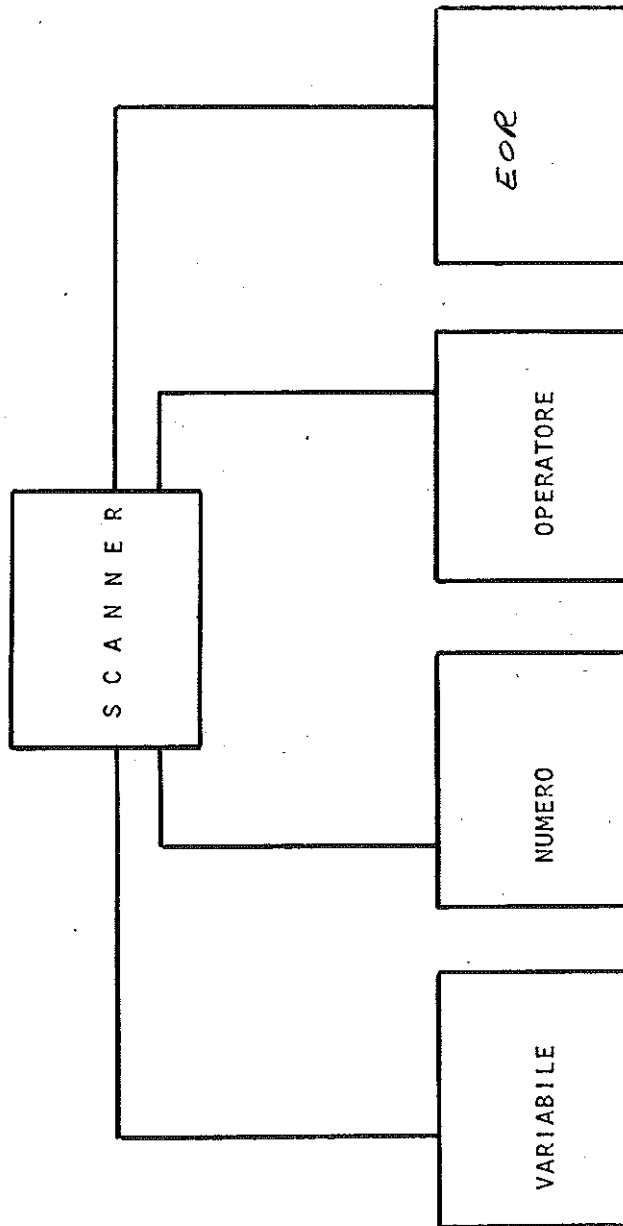



fig. 3.2.1

input e che l'operatore può controllare la macchina.

Un programma è costituito da una o più linee (records) di lunghezza variabile. Ogni linea può essere identificata da una etichetta costituita da un qualunque carattere (con esclusione dei caratteri CR,";", BELL) preceduto dall'operatore LABEL ("[") . Le linee non devono necessariamente essere etichettate e un'etichetta può trovarsi in qualunque punto del programma, anche più di una in ogni linea.

Ogni linea termina col carattere "End of record" (ritorno carrello). Un programma deve terminare col carattere "End of file" (%).

4.1 - Comandi di sistema

Ogni comando di sistema deve essere preceduto dal carattere ";" che produce eco "(".

Subito dopo aver battuto il comando viene automaticamente stampato il carattere ")" in modo da facilitare l'individuazione dei comandi sullo stampato che risultano racchiusi tra parentesi.

All'atto della inizializzazione viene automaticamente inserito un carattere EOF (%) sulla prima locazione di

memoria programma. Tutti i caratteri che verranno da qui battuti, con esclusione dei comandi, saranno inseriti nella memoria programma a partire dalla prima locazione.

Un puntatore (pointer) consente di effettuare tutte le operazioni di editing, listing ed esecuzione del programma.

Tutti i comandi indicati possono essere impartiti indifferentemente con lettere maiuscole o minuscole.

In caso di comando non conosciuto l'esecuzione riprende dalla locazione H'8080' (Debug).

4.1.1 Comando @

Serve per tornare sotto il controllo del Debug. La macchina stampa pertanto "?".

4.1.2 Comando B (Beginning of line)

Riporta il puntatore all'inizio della linea che si sta attualmente battendo. Durante l'introduzione del programma può essere usato per cancellare la linea in corso di battitura.

4.1.3 Comando D (Decrement)

Sposta il puntatore indietro di un posto.

4.1.4 Comando E (Execute)

Permette di eseguire un programma precedentemente introdotto in memoria. L'esecuzione inizia dalla posizione attuale del pointer. Nel caso che il programma termini con l'esecuzione di un operatore EOF (%) il controllo viene restituito all'operatore. Diversamente è necessario riinizializzare l'esecuzione e la macchina stamperà di nuovo l'intestazione. In questo caso, come si è già visto, viene inserito automaticamente un carattere % nella prima locazione di memoria programma. E' quindi necessario, prima di effettuare le operazioni di editing sul programma già in macchina, ripristinare nella prima locazione il carattere con cui inizia il programma. I contenuti delle variabili non sono alterati dalla riinizializzazione. Diversamente si può inizializzare l'interprete dalla locazione H'1035': il programma non verrà alterato.

4.1.5 Comando F (Find)

Il comando FIND serve per posizionare il puntatore su una certa locazione di memoria programma in base ai due caratteri che la precedono. La ricerca avviene a partire dalla posizione attuale del pointer. Nel caso che la ricerca abbia un esito negativo verrà stampata la scritta NOT FND ed il pointer viene riportato sulla prima locazione di memoria programma. Esempio:

```
* R P N / 8 A  (Vers.28.08.77)

:"Linea di priva"%(h)
:(f)pr
:o(h)
:(c)
"Linea di prova"%
:
```

Ecco come correggere una linea sbagliata:

1) Si batte un programma e alla fine ci si accorge di aver sbagliato un carattere;

2) Si posiziona il pointer sulla prima locazione col comando HOME (vedi)

3) Si ricerca col comando FIND, la sequenza "pr". Il pointer si posiziona sulla locazione successiva che contiene una i

4) Si batte una "o" che va a sostituire la i.

5);6),7) La linea è stata corretta come si può osservare con il comando LIST (vedi)

4.1.6 Comando G (Give)

Viene stampato il carattere puntato dal pointer ; quest'ultimo non risulta modificato.

4.1.7 Comando H (Home)

Il puntatore viene riportato sulla prima locazione della memoria programma.

4.1.8 Comando N (Next)

Sposta il puntatore avanti di un posto.

4.1.9 Comando C (Compact List)

Viene stampata la lista del programma a partire dalla locazione attuale del pointer.

4.2 Le variabili

L'RPN/8A riconosce 16 variabili identificate da una delle prime lettere dell'alfabeto inglese e dal carattere @ . Ogni variabile può contenere un numero di 14 cifre significative, segno ed esponente con segno.

4.3 Lo stack

Le operazioni aritmetiche vengono eseguite su una struttura a pila, nel seguito denominata stack.

Lo stack può contenere 4 quantità numeriche, sempre con 14 cifre significative, segno, esponente col segno.

Una costante od una variabile può essere introdotta nello stack semplicemente facendola seguire dall'operatore di ENTER (vedi). Ad esempio, la scrittura:

123 2 B 41.7

fa si che lo stack venga a contenere 123,2,B,41.7.

Il numero 41.7 si trova nel registro più basso dello stack (denominato "X") e 123 in quello più alto (denominato "T").

fig. 4.3.1.a

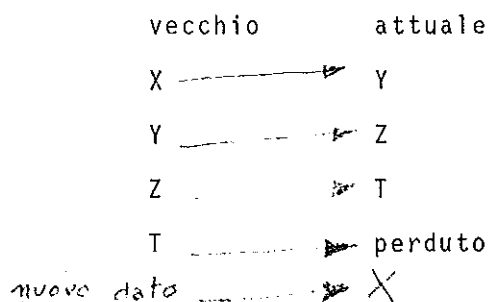
vecchio		attuale
X	→	Y
Y	→	Z
Z	→	T
T	→	perduto
nuovo dato	→	X

I dati che entrano nello stack vengono posti nel registro X e lo stack "sale" di un posto. Il contenuto del registro T viene perduto.

fig. 4.3.1.b

vecchio	attuale
X	T(di solito)
Y	X
Z	Y
T	Z

I dati che "escono dallo stack vengono prelevati dal registro X mentre lo stack "scende"di un posto. Il vecchio Y diventa l'attuale X, Z diventa Y, T diventa Z. Il nuovo T assume di solito lo stesso valore del vecchio X.

123 2 B 41.7 fa sí che lo stack li contenga secondo lo schema di fig. 4.3.2.

fig.4.3.2

123	T
2	Z
B	Y
41.7	X

Il caricamento avviene in questi 4 tempi:

Inizio		1° tempo		2° tempo	
X contiene		123	X	123	Y
Y " " "		Y contiene		2	X
Z " " "		Z " " "		Z contiene	
T " " "		T " " "		T " " "	

= contenuto casuale

3° tempo		4° tempo	
123	Z	41.7	X
2	Y	B	Y
B	X	2	Z
T contiene		123	T

Le operazioni diadiche (ad esempio: prodotto) vengono eseguite sulla coppia di registri X ed Y. Le operazioni monadiche (ad esempio valore assoluto) vengono eseguite sul registroX.

4.4 Gli operatori

4.4.1 L'operatore di ENTER

L'operatore di ENTER è costituito dal carattere spazio.

Effetto: il numero o la variabile che precede l'operatore di ENTER viene posto nel registro X e lo stack "sale" di un posto.

Nota: L'operatore ENTER è necessario solo fra due numeri; in caso contrario può essere omesso. Ad esempio:

A A può essere scritto AA .

Mai:

123 456 non può essere ovviamente scritto: 123456.

4.4.2 L'operatore GOTO (!)

Effetto: il controllo passa alla prima locazione succes-

siva all'etichetta indicata dopo l'operatore GOTO; lo stack non risulta alterato.

Esempio: !3 significa: vai ad eseguire dopo l'etichetta 3

!A " " " " " " " " " " " " " " A

4.4.3 L'operatore TEXT (")

Effetto: tutto ciò che segue l'operatore TEXT viene direttamente ristampato in fase di esecuzione finchè non si incontra un successivo operatore TEXT; successivamente riprende la normale elaborazione. Lo stack non viene alterato.

Esempio: "PIPPO PLU""TO" viene stampato PIPPO PLUTO.

4.4.4 L'operatore FIX (f)

Effetto: tramite l'operatore FIX é possibile prefissare il numero dei decimali stampati per mezzo dell'operatore PRINT x, facendo seguire al medesimo, senza spazi interposti, una cifra da 0 a 9 ed una delle lettere J,K,L,M che rappresentano rispettivamente 11,12,13,14 cifre decimali. Lo stack non viene alterato.

Esempio: f5 prefissa una stampa con 5 cifre decimali

fj " " " " " " " " 11 " " " "

La precisione dei calcoli non viene alterata dallo operatore FIX e rimane sempre la stessa consentita.

4.4.5 L'operatore PRINT (#)

Effetto: il contenuto del registro X viene stampato secondo il formato prefissato dall'operatore FIX. Alla fine della stampa viene automaticamente inserito uno spazio. Nel formato FIX0 non possono essere stampate più di 14 cifre prima del punto decimale. Nel formato FIX1; 13 e così via; non ci sono invece limitazioni per la stampa di numeri senza cifre significative a sinistra del punto decimale, (número in valore assoluto 1) se non quelle derivanti dalla rappresentazione interna dei numeri che ha sempre esponente maggiore di 10^{-125}.

Esempio: f3 431.56749# ha effetto:

431.567

4.4.6 L'operatore DOLLARO ($)

Effetto: serve per il controllo della stampante e provoca un ritorno carrello/interlinea.

Esempio: "PLUTO"$"PAPERINO" viene stampato: come:

PLUTO

PAPERINO

4.4.7 L'operatore END ("%")

Effetto: il controllo viene restituito all'operatore. Deve essere presente in ogni programma per arrestare l'esecuzione e per indicare la fine fisica del programma stesso.

4.4.8 L'operatore STO (&)

Effetto: Il contenuto del registro X viene assegnato alla variabile che di regola deve seguire l'operatore STO senza spazi interposti. Lo stack non viene modificato.
Esempio: 23.4 &A fa assumere alla variabile A il valore: 23.4.

4.4.9 L'operatore MULTIPLY (*)

Effetto: viene eseguito il prodotto dei contenuti dei registri X ed Y. Lo stack scende di un posto. In caso di fuori capacità si torna in Debug.
Esempio: 2 43.1* viene stampato: 86.20

4.4.10 L'operatore SUM (+)

Effetto: Viene eseguita la somma dei contenuti dei registri X ed Y. Lo stack "scende" di un posto. In caso di fuori capacità si torna in debug.
Esempio: 2 57 + viene stampato: 59.00

4.4.11 L'operatore SUB (-)

Effetto: il contenuto del registro X viene sottratto dal contenuto del registro Y. Lo stack "scende" di un posto. In caso di fuori capacità si torna in debug.
Esempio: 400 32- stampa: 368.00

4.4.12 L'operatore DIVIDE (/)

Effetto: il contenuto del registro Y viene diviso per il contenuto del registro X. Lo stack "scende" di un posto. Nel caso di fuori capacità o di divisione per Zero si torna in debug.
Esempio: 5 2/ stampa: 2.50

4.4.13 L'operatore SEPARAZIONE (,)

Effetto: Nessuno. Serve solo per chiarezza tipografica,

per separare vari elementi di una frase. Allo stesso scopo può essere usato il carattere spazio, tranne in quei casi in cui ciò sia esplicitamente escluso. E' utile inserire SEPARAZIONI e spazi anche per aggiungere eventualmente nuovi elementi in un programma senza dovere rintrodurre il medesimo in memoria.

Esempio: A31.4 5+ può essere scritto: A 31.4 ,5+

4.4.14 L'operatore ROLL DOWN (R)

effetto: fa scendere di un posto lo stack come indicato da questo schema:

vecchio	attuale
X	T
Y	X
Z	Y
T	Z

Esempio: 4 5 78 11 #, R#, R#, R# ha effetto: 11 78 5 4 .

4.4.15 L'operatore ABS (a)

effetto: il contenuto del registro X vien e sostituito con il valore assoluto del contenuto medesimo. Lo stack non viene alterato.

Esempio: 4_ a   viene stampato 4.

4.4.16 L'operatore CHS (_)

Effetto: il segno del contenuto del registro X viene cambiato. Lo stack non viene alterato.

Esempio: 3 4 *_ha effetto: -12.00

4.4.17 L'operatore JMP (ȧ)

Effetto: il controllo viene passato alla locazione il cui indirizzo, inteso come numero esadecimale, si trova nel registro X dello stack. Poiché nello stack non possono trovarsi che quantità numeriche non si possono effettuare salti ad indirizzi che contengono lettere. Lo stack non viene alterato.

Esempio: 8080 ȧ salto alla locazione H'8080'

8306 ȧ non ammesso.

Nota: devono sempre essere specificate almeno 4 cifre, la prima delle quali diversa da zero.

4.4.18 L'operatore di INPUT (i)

Effetto: L'elaborazione viene interrotta ed un suono di campanello avverte l'operatore che la macchina si dispone ad accettare una quantità numerica. Il numero battuto viene introdotto nel registro X quando si preme il ritorno carrello (numero positivo) o il carattere "_" (numero negativo). In caso di errore si possono cancellare i caratteri battuti con il carattere BELL (eco "\") tante volte quanto necessario. (Ad ogni battuta viene cancellato un carattere.)
Eesempio: "Aliquota Iva" i & I ha per effetto che il numero battuto in risposta al suono di campanello viene assegnato alla variabile I.

4.4.19 L'operatore REPLY (r)

Effetto: L'elaborazione viene interrotta ed un suono di campanello avverte l'operatore che la macchina si dispone ad accettare dalla tastiera un ingresso alfanumerico per ristamparlo fedelmente sul foglio di carta della stampante. L'elaborazione riprende alla pressione del tasto ritorno carrello.

Esempio: 33 r "PIPPO" ha effetto:

33 (suono campanello) Commento bla,bla,bla (CR) PIPPO

4.4.20 L'operatore IF X.EQ.0 (z)

SPAZIO INTENZIONALMENTE BIANCO

Effetto: Viene controllato il registro X. Si hanno due casi:

a) Se x = 0 l'esecuzione prosegue normalmente e l'operatore IF X.EQ.0 non ha alcun effetto.

b) Se X ≠ 0 l'esecuzione prosegue dalla locazione successiva al primo End Of Record (ritorno c arrello) che segue l'operatore IF X.EQ.0.

Lo stack non viene alterato.

Esempio: A z"PIPPO"
"PLUTO"

ha effetto: PIPPO PLUTO se X = 0

ha effetto PLUTO se X ≠ 0

4.4.21 L'operatore IF X.EQ.Y (e)

Effetto: viene controllata l'identità tra i registri X ed Y;si hanno due casi:

a) Se X = Y L'esecuzione procede normalmente e l'operatore IFX.EQ.Y non ha alcun effetto.

b) Se X ≠ Y l'esecuzione procede dalla locazione successiva al primo End Of Record (ritorno carrello) che segue lo operatore IFX.EQ.Y

Lo stack esegue due ROLL DOWN (vedi)

AVVERTENZA IMPORTANTE - A seguito della organizzazione della aritmetica interna e' possibile che alcuni calcoli diano risultati in forma non normalizzata. Poiche' l'operatore IF X.EQ.Y opera sulla identita' formale dei due registri X ed Y, quando il contenuto di uno di essi (o di entrambi) sia risultato di operazioni aritmetiche e' opportuno normalizzarne la forma tramite la divisione per l'unita'.

ESEMPIO:

```
* R P N / 8 A1 (Vers.28.08.77)

:2.5 STO A
:AAA** AA*- "Risultato non normalizzato " PRINT
:"   Risultato normalizzato " 1/ PRINT
:END (h)
:(e)Risultato non normalizzato 09.37    Risultato normalizzato 9.37
:
```

4.4.22 L'operatore IF X.LT.0 (1)

Effetto: viene controllato il segno del registro X.
Lo stack scende di un posto. Si hanno due casi:

a) X < 0 l'esecuzione continua normalmente.

b) X ≥ 0 l' esecuzione continua dalla locazione successiva al primo End Of Record (ritorno carrello) che segue l'operatore IF X.LT.0.

Lo stack esegue un ROLL DOWN (vedi).

Avvertenza: in conseguenza della particolare struttura della sezione aritmetica dell'interprete è possibile che alcuni calcoli abbiano come risultato -0.00 invece di 0.00. L'operatore IF X.LT.0 deve essere pertanto usato con le dovute cautele, ad esempio facendolo precedere dagli operatori IF X.EQ.0 ABS.

4.4.23 L'operatore TAB(T)

Effetto: vengono stampati tanti caratteri spazio (" ") tante sono le unità del registro X. In caso di numero non intero si effettua l'arrotondamento all'intero superiore. L'effetto è lo stesso per numeri positivi o negativi.

Nota: vengono alterati tanto lo stack quanto il contenuto della variabile @ .

Esempio: 13.6 T " " ha effetto: 14 spazi

APPENDICE A

I comandi dell' RPN/8A :

@ - Ritorno al DEBUG

b - Beginning of line

c - Compact list

d - Decrement pointer

e - Execute

f - Find

g - Give (stampa carattere attuale)

h - Home

n - Next

Gli operatori dell'RPN/8A :

Spazio - Enter

! - Go to

" - Text

- Print

$ - CR/LF

% - End

& - Store

* - Multiply

+ - Sum

l - IF X.LT.0

e - IF X.EQ.Y

- - Sub

/ - Divide

, - Separazione

R - Roll down

a - Valore assoluto

_ - CHS

j - JMP

i - Input

r - Reply

z - IF X.EQ.0

T - Tab

[- LRL

f - Fix

APPENDICE B

Occupazione di memoria dell'RPN/8A: 2K a partire dal la locazione H'1ØØØ', piu' un K RAM dalla loc. zero.

Entry points: Inizio: H'1ØØØ'

Inizio senza distruzione del programma utente: H'1Ø35'.

Hardware richiesto: 1 scheda CHILD 8/BS CPU

1 " CHILD 8/BS SMB

1 " telescrivente ASCII con full set di 96 caratteri

(in alternativa alla scheda SMB puo' essere usata una scheda PROMB con interprete RPN/8A. in ROM, cod.10003R)

Ø4Ø18

I Edizione

Gianni Becattini

RPN/8A1

MANUALE DI UTENZA

Il presente manuale, complementare al GP-Ø4Ø17, comprende le informazioni aggjuntive sulla versione A1 dell'interprete ad alto livello RPN/8A

SISTEMI DI ELABORAZIONE - MICROPROCESSORI
VIA MONTEBELLO, 3 - 3a rosso
TEL. 055 / 219.143 — 50123 FIRENZE

1-Principali differenze con la versione A

Molto importante e' la differenza tra la versione A e la A1 di cui sono oggetto queste note. L'innovazione essenziale e' costituita dalla cosiddetta "precompilazione" o caratteristica di "list esteso". In base ad essa e' possibile avere una stampa in chiaro del codice mnemonico corrispondente ad ogni operatore. Ad esempio, l'operatore corrispondente al tasto '#' viene stampato come'PRINT'. In questo modo sono stati raggiunti i seguenti importanti risultati:

1) I programmi sono facilmente "leggibili" grazie allo uso esteso di codici mnemonici intuitivi.

2) L'occupazione di memoria rispetto ad altri linguaggi che memorizzano i codici mnemonici per esteso e' inferiore di circa l'80% in molti casi.

3) La velocita' di esecuzione e' in molti casi pari alla riduzione di occupazione di memoria del programma utente. Difatti non e' necessario ritradurre i codici in fase di esecuzione.

4) Battere un programma diventa molto piu' semplice e piu' veloce in virtu' del numero limitato di pressioni di tasto richieste; ad esempio per l'operatore

IF X.LT.0 (9 caratteri) e' richiesta una sola pressione di tasto invece che 9.

Oltre alla precompilazione sono stati aggiunti tre nuovi comandi, utilissimi per la correzione dei programmi, in particolare l'insert ed il delete che consentono di inserire o cancellare un carattere dalla memoria spostando tutto il programma gia' esistente in modo da ottimizzare la occupazione di memoria.

Sono stati aggiunti anche 10 nuovi operatori di notevole importanza ed un registro W utile per molte applicazioni.

2 - Comandi di sistema

Sono stati aggiunti i seguenti comandi:

2.1 - Comando L (list)

Serve per ottenere la lista estesa del programma, a partire dalla locazione attuale del pointer. L'operazione termina all'operatore END.

2.2 - Comando K (kill)

Viene cancellato dalla memoria il carattere cui punta il pointer. Tutto il prpgramma utente che segue l'operatore eliminato viene fatto "risalire" di un posto a colmare la locazione libera. Prima di impartire il comando K e' bene assicurarsi della posizione del pointer con il comando G (V.).

2.3 - Comando I (insert)

Viene inserito un nuvo carattere subito prima della posizione attuale del pointer. Tutto il programma utente che segue e' fatto scorrere di una locazione per dare spazio al nuovo carattere. Esempio:

```
:(I)PRINT
:
```

L'operatore PRINT (tasto #) viene inserito prima della locazione attuale del pointer.

3 - I nuovi operatori

Si riportano nel seguito i nuovi operatori aggiunti rispetto alla versione A. Ogni operatore e' in-

dicato con la lettera del tasto corrispondente tra parentesi.

3-1 - <u>L'operatore CALL</u> (c)

<u>Effetto</u>: L'operatore CALL deve essere seguito, senza aggiungere spazi, da una etichetta. All'atto della esecuzione il controllo passa al sottoprogramma identificato dalla etichetta indicata.
Un sottoprogramma non puo' contenere gli operatori PSH, VAR, STK, TAB, CALL. Ogni sottoprogramma deve terminare con l'operatore RETURN.

3.2 - <u>L'operatore RETURN</u> (P)

<u>Effetto</u>: Il controllo ritorna al programma principale, alla locazione subito successiva agli operatori CALL <etichetta> eseguiti per ultimi.

3.3 - <u>L'operatore STOP</u> (S)

<u>Effetto</u>: L'esecuzione del programma si arresta ed il controllo torna all'operatore (la macchina stampa i

due punti). A differenza dell'END, che indica anche la fine fisica del programma, lo STOP puo' trovarsi in qualunque posto del programma stesso.

Esempio:

```
IF X.EQ.Y $ "EGUAGLIANZA" STOP
3 4* 5/ A+
...........
END
```

3 .6 - L'operatore STK (s)

Eff.: Viene stampato il contenuto dello stack che a sua volta non risulta alterato.

Esempio di esecuzione dell'operatore STK:

```
X: 3.14
Y: 4.51
Z: 1.23
T: 3.33
```

3.7 - L'operatore VAR (V)

Effetto: Vengono stampati, in forma analoga a quella vista per l'operatore STK, i contenuti delle variabili.

3.8 - L'operatore RU (U)

Effetto: Esattamente opposto a quello dell'operatore RD. Lo stack viene fatto ruotare di un posto verso lo alto.

3.9 - L'operatore USER (u)

Effetto: Il controllo passa alla locazione di memoria assoluta H'ØØBF'. Un eventuale ritorno allo scanner RPN/8A1 puo' essere fatto con un salto alla locazione H'12FØ'.

3.10 - L'operatore X()Y (Y)

Effetto: Vengono scambiati i contenuti dei registri X ed Y.

3.11 - L'operatore X()W (X)
Effetto: Vengono scambiati i contenuti dei registri X e W.

3.12 - L'operatore PSH (p)

Effetto: Il contenuto del registro X viene ricopiato in Y. Il vecchio contenuto di T viene perduto e lo stack sale di un posto.

AVVERTENZA IMPORTANTE - Nello spazio tra due operatori TEXT ("), non possono essere impartiti comandi ne'si possono effettuare operazioni di correzione o di editing. La stringa ricercata con il comando find non deve contenere il carattere ".

APPENDICE A

I comandi aggiunti nella versione A1:

I - Insert

K - Kill

L - list

Gli operatori

Operatore	Tasto da premere
GOTO	!
"	"
PRINT	#
$	$
END	%
STO	&
*	*
+	+
IF X.LT.0	l
IF X.EQ.0	z
-	-
/	/
,	,
FIX	f
RD	R
ABS	a
CHS	_

CHS	_
JMP	j
INPUT	i
REPLY	r
IF X.EQ.Y	e
TAB	T
CALL	c
RETURN	P
STOP	S
STK	s
VAR	V
RU	U
USER	u
X()Y	Y
X()W	X
PSH	p
LBL	[

APPENDICE B

Occupazione dell'RPN/8A1: 2,5 K di memoria a partire dalla locazione H'1ØØØ' RAM o ROM, piu' un K di RAM a partire dalla locazione H'ØØØ' (questo K puo' essere espanso fino a 4 K complessivi, sempre a partire da zero. Con un K solo si hanno a disposizione circa 760 passi di programma utente. L'aggiunta dei K successivi permette di raggiungere i 3840 passi complessivi..

Entry points: vedi RPN/8A

Vedi RPN/8a. Il codice della versione RPN/8A1 su ROM e' 10004R.

Ø4Ø19

I EDIZIONE

Gianni Becattini

MANUALE DI UTENZA
=================================

Il presente manuale, complementare ai GP-Ø4Ø17 e GP-Ø4Ø18, comprende le informazioni aggiuntive sulla versione A2 dell'interprete ad alto livello RPN/8A.

SISTEMI DI ELABORAZIONE - MICROPROCESSORI
VIA MONTEBELLO, 3 - 3a rosso
TEL. 055 / 219.143 — 50123 FIRENZE

1 - Principali differenze rispetto alla versione A1

La principale innovazione rispetto alla versione A1 e' costituita dalla creazione di nuovi operatori per un migliore controllo della tabulazione sia orizzontale che verticale nonche' per espandere lo spazio a disposizione per i dati numerici. Un sistema di gestione totalmente automatico alloca un vettore di variabili R(W) nello spazio di memoria RAM lasciato libero dal programma. Questo permette al programmatore di creare programmi piu' brevi con maggiore quantita' di dati da trattare o viceversa a seconda delle necessita'. E' stato aggiunto un nuovo registro detto "line register" per il conteggio delle linee stampate e per la tabulazione.

2 - I nuovi operatori

Sono stati aggiunti complessivamente 6 operatori. Sono indicati anche qui con il loro nome, seguiti, in parentesi, dal tasto corrispondente.

2.1 - L'operatore LINES (x)

Effetto: Il contenuto del registro W viene trasferito nel line register. I numeri devono essere compresi nel rango 0-999.
Esempio: per caricare 12 nel line register si opera cosi':
12 X()W LINES

2.2 - L'operatore $ ($)

Effetto: L'operatore $ della versione A2 differisce da quello incontrato nelle versioni precedenti. Difatti adesso, oltre alla stampa del CR/LF, viene anche decrementato il line register.

2.3 - L'operatore FORM (w)

Effetto: Vengono stampati tanti CR/LF quante sono le unita' del line register. E' chiamato anche "operatore di pagina nuova".

2.4 - L'operatore TAB (W) (W)

Effetto: Vengono stampati tanti caratteri spazio tante

sono le unita' del registro W, MENO il numero delle cifre a sinistra del punto decimale del numero contenuto nel registro X. Se ad esempio X contiene 123456.1 e W 10, vengono stampati solo 10-6=4 spazi. Cio' e' particolarmente utile quando si effettuano operazioni di incolonnamento.

2.5 - L'operatore STO R(W) - (g)

Effetto: Il contenuto del registro X e' ricopiato nel registro per dati di indice W. Per immettere un numero nel registro R(W) numero 4 si opera cosi':

4 X()W 998.76 STO R(W)

Il numero dei registri R(W) a disposizione e' dato da

(M-L)/9

(circa), dove M=capacita' della memoria utente espressa in passi di programma possibili, L=lunghezza del programma utente al momento della esecuzione, sempre in numero di passi.

2.6 - L'operatore RCL R(W) (h)

Effetto: Il contenuto del registro R(W) indirizzato dal registro W e' copiato in X. Lo stack sale di un posto.

2.7 - L'operatore CALL (c)

A differenza delle versioni precedenti e' possibile usare qualsiasi operatore nel sottoprogramma tranne un altro CALL.

APPENDICE A

Operatori aggiunti nella vers.A2

Operatore	Tasto corrispondente
LINES	x
FORM	w
$	$
CALL	c
STO R(W)	g
RCL R(W)	h
TAB (W)	W

APPENDICE B

Occupazione di memoria: 3.0 K di memoria RAM o ROM (vedi vers.A1 per la memoria utente)

Entry points: vedi vers.A1

Hardware: Vedi versioni precedenti. Il codice dello interprete su ROM e' 10005R.

L' USO CON LA

(od altra telescrivente in codice Baudot)

1 - Il collegamento fisico e l'hardware necessario

L'impiego di una telescrivente in codice Baudot, assai facilmente reperibile nel mercato del surplus, consente un impiego estesissimo della elaborazione elettronica RPN/8A anche in ambienti del tutto differenti dai tradizionali. Difatti con una spesa inferiore a quella necessaria per l'acquisto di una media stazione di radioamatore ognuno puo' costruire da solo ed in poco tempo un piccolo "centro di elaborazione dati" casalingo con possibilita' estesissime anche su livello professionale.

Due sono le possibilita' di operare, in dipendenza del tipo di memoria usata per memorizzare lo interprete RPN/8A2T. La prima si basa sull'utilizzo di memoria ROM, fornita gia' programmata dalla General Processor stessa, la seconda prevede invece l'uso di memoria RAM, caricata tramite mangiacassette od altra unita' di ingresso uscita.

Al momento in cui sono scritte queste note non e' ancora disponibile l'interfaccia per mangiacassette, peraltro assai facilmente realizzabile sulla base di quanto esposto sul numero 3/77 di HOB-BIT e quindi

neppure le relative cassette (e' previsto comunque un tempo di attesa non troppo lungo).

Disponendo di una cassetta di tipo AUTO-LOAD e' sufficiente accendere il registratore in riproduzione per caricare l'interprete RPN/8A2T nella scheda SMB da 4 K RAM. Chi invece lo desidera potra' optare per l'uso di memoria ROM fornita gia' programmata. Per conseguire la massima facilita' di impiego e' necessario operare una semplicissima modifica sulla scheda CPU (tale modifica e' descritta dettagliatamente nel seguito).

Il collegamento con la telescrivente avviene tramite l'interfaccia General Processor 09000 (da alimentare a parte con un trasformatore a 9V e 80 V di secondario), secondo le istruzioni allegate alla interfaccia stessa, occupando 4 bit del port di I/O numero 5. L'interfaccia 09000 aggiunge alla telescrivente 2 tasti di funzione, il FIGS, che risparmia il fastidioso compito di manovrare i tasti FIGS/LTRS che gia' ci sono nella telescrivente (si usa come il maiuscolo delle macchine da scrivere, da premersi CONTEMPORANEAMENTE al tasto voluto), ed il tasto CONTROL la cui funzione sara' spiegata parlando della program

zione dell'RPN/8A2T.

Riassumiamo qui, per la comodita' del lettore, le due configurazioni minime richieste per far funzionare l'RPN/8A2T con la telescrivente Baudot:

Sistema con mem. RAM	Sist. con m. ROM
Scheda CHILD 8/BS CPU	Idem
" SMB da 4K RAM	Scheda PROMB
-	Set di ROM GP-10006
Interfaccia 09000	Idem
Interfaccia cassette mod.	
GP-11000 (disp. pross.)	(Idem opzionale)
Alimentaz. ed interconn.	Idem

2 - Le differenze con l'RPN/8A2

Per permettere l'adattamento sulla telescrivente Baudot, sono state operate sull'RPN/8A2T alcune modifiche e miglioramenti. Sono stati aggiunti due

nuovi comandi per salvare su cassetta o per carica-
re, programmi utente in RPN/8A2T. Oltre a cio' si'
e' prevista la funzione BREAK, per interrompere in
qualsiasi istante la elaborazione e tornare sotto
il controllo del sistema operativo (quando la mac-
china stampa i due punti).

Tutti i numerosi operatori dell'RPN/8A2T
si ottengono dalla normale tastiera della TG7 (o
altra telescrivente Baudot) dai normali tasti combi-
nandone la pressione con quella di uno dei tasti
FIGS e CONTROL dell'interfaccia 09000.

Vediamo le differenze rispetto alle versio-
ni ASCII.

Carattere od operatore	diviene
@	' (apice)
# (PRINT)	(
% (END)	)
+	CONTROL/D (Eco "SUM")
_ (CHS)	CONTROL/K
*	CONTROL/M (Eco "MUL")
[(LBL)	CONTROL/O

Tutti i tasti ASCII minuscoli si ottengono come CONTROL/
/lettera desiderata.

IMPORTANTE: I tasti LTRS e BLANK non devono essere usati. Il vecchio tasto FIGS serve come correzione dei caratteri battuti con eco "?", al posto della barra contraria. I nuovi tasti FIGS e CONTROL devono essere usati come il "Maiuscolo" della macchina da scrivere. Ad esempio, per ottenere il numero 3 si fa cosi':

1) Premere e tenere premuto il tasto FIGS
2) Battere il tasto (3 E)
3) Lasciare il tasto FIGS

3 - I nuovi comandi

I nuovi comandi sono:

3.1 - Il comando Output memory (.O)

Tutto il programma utente, a partire dalla locazione attuale del pointer, viene emesso in codice ASCII verso il nastro magnetico. La velocita' di emissione é di 300 Baud.

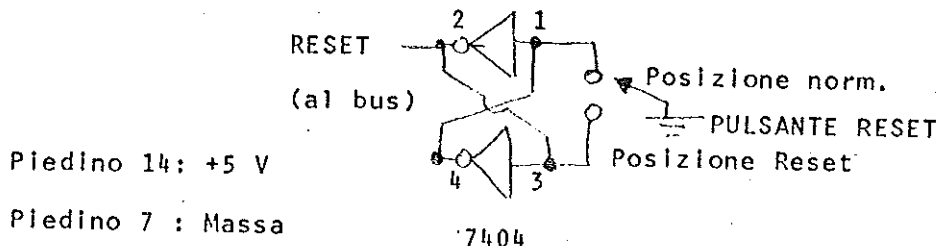
3.2 - Il comando Memory load (M)

La memoria utente, a partire dalla prima locazione, viene caricata con il programma utente letto da nastro alla velocita' di 300 baud. Eventuali LF, RUBOUT, NULL sono ignorati.

4 - Le modifiche alla CPU

Le modifiche da apportare alla scheda CPU sono estremamente semplici.

1) Inserire, in serie alla pista che va dal pin 6 del 4075 ai pin 13 delle 2102 e al pin 13 del 7406, un interruttore a pulsante normalmente chiuso.

2) Collegare il pulsante di reset al bus tramite il circuito antirimbalzo della figura seguente (*).

(*) - Opzionale ma consigliato

Per iniziare l'esecuzione dell'RPN/8A2T si opera poi cosi': (per interprete in ROM)

1) Accendere il microcomputer

2) Porre lo switch DEBUG/PGM in posizione PGM

3) Premere il pulsante aggiunto, il cui nome e' LOAD, e tenerlo premuto.

4) Premere il pulsante RESET.

5) Rilasciare il pulsante LOAD

6) Battere un tasto qualunque; la macchina stampa la scritta

R P N /8A2T(Vers.xx.yy.zz)

dove xx.yy.zz e' la data della versione in corso.

5 - La funzione BREAK

E' possibile interrompere in qualsiasi istante l'esecuzione del programma premendo il pulsante CONTROL sull'interfaccia 9000. La macchina stampa allora i due punti in attesa di ordini.

FOGLIO AGGIUNTIVO
''''''''''''''''''''''''''''''''''

Tutte le versioni di RPN/8A2T successive alla 30.09.77 sono dotate del nuovo oomando "A". Tramite esso e' possibile stampare tutta la linea su cui e' posizionato il pointer fino al primo CR (compreso. Cio' e' di grande utilita' per la correzione e verifica dei programmi.. Dopo l'esecuzione del comando A il pointer resta sul primo elemento della riga successiva.